M1
SIGLIS

ANTOINE Stéphane

DUPOY Vincent

HASSAN DJABER Mohamed

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

[LA SOCKET CALCULATRICE]

Ingénierie des réseaux - Projet Socket

DOSSIER DE PROGRAMMATION

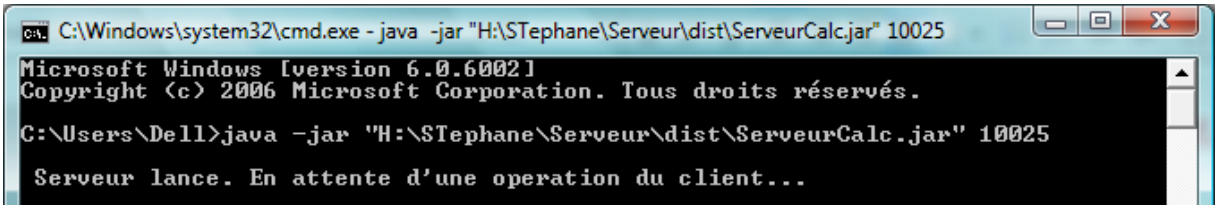
1. Mode d'emploi p.3
2. Analyse p.4
3. Conception p.5
4. Problème de Développement p.6
5. Bilan p.7

1. Mode d'emploi

- Dans une Invite de commande lancer « ServeurCalc.jar » à l'aide de la commande :

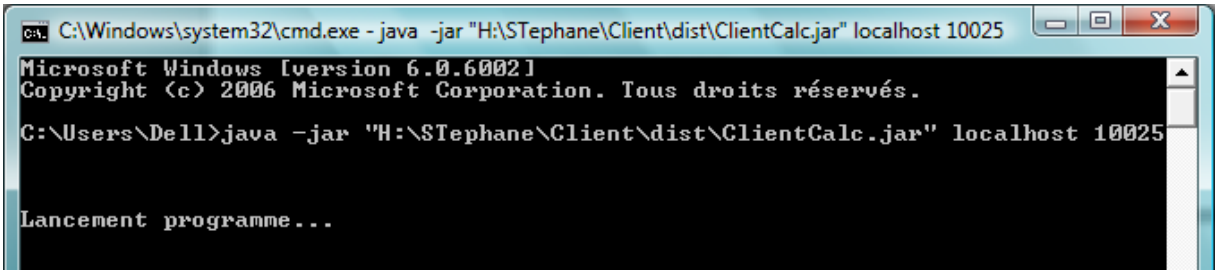
java –jar « *chemin d'accès au fichier ServeurCalc.jar* » *num_port*

Un message atteste que le serveur est bien lancé. Il est recommandé de ne pas utiliser les ports réservés (plage 0 à 1023) sinon Windows vous demandera de confirmer.

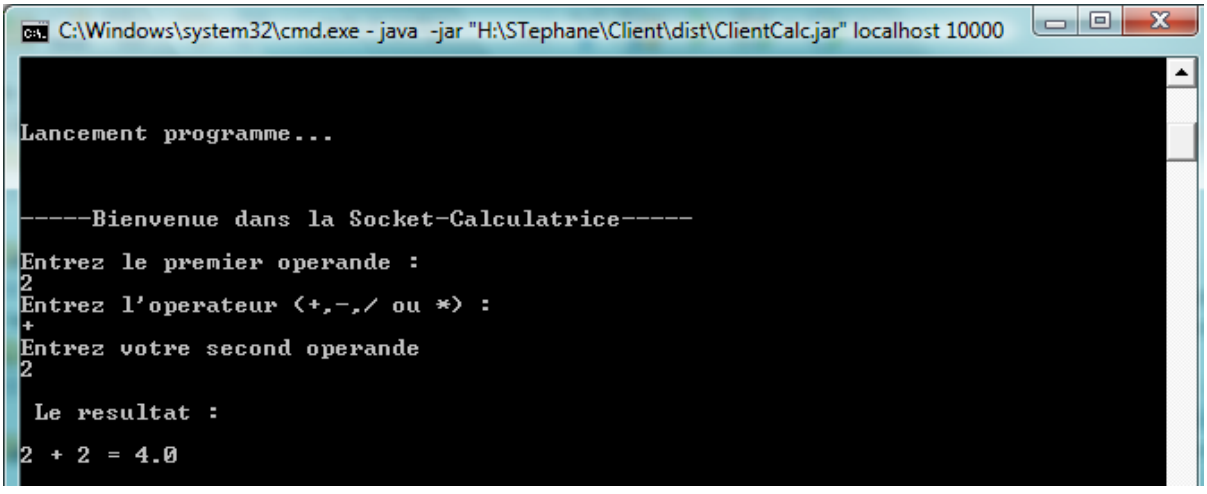

C:\Windows\system32\cmd.exe - java -jar "H:\STephane\Serveur\dist\ServeurCalc.jar" 10025
Microsoft Windows [version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation. Tous droits réservés.
C:\Users\Dell>java -jar "H:\STephane\Serveur\dist\ServeurCalc.jar" 10025
Serveur lance. En attente d'une operation du client...

- Dans **<u>une nouvelle</u>** invite de commande, lancer le client à l'aide de la commande :

java –jar « *chemin d'accès au fichier ClientCalc.jar* » *adrIPduServeur num_port*


C:\Windows\system32\cmd.exe - java -jar "H:\STephane\Client\dist\ClientCalc.jar" localhost 10025
Microsoft Windows [version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation. Tous droits réservés.
C:\Users\Dell>java -jar "H:\STephane\Client\dist\ClientCalc.jar" localhost 10025
Lancement programme...

- Suivre les instructions du programme

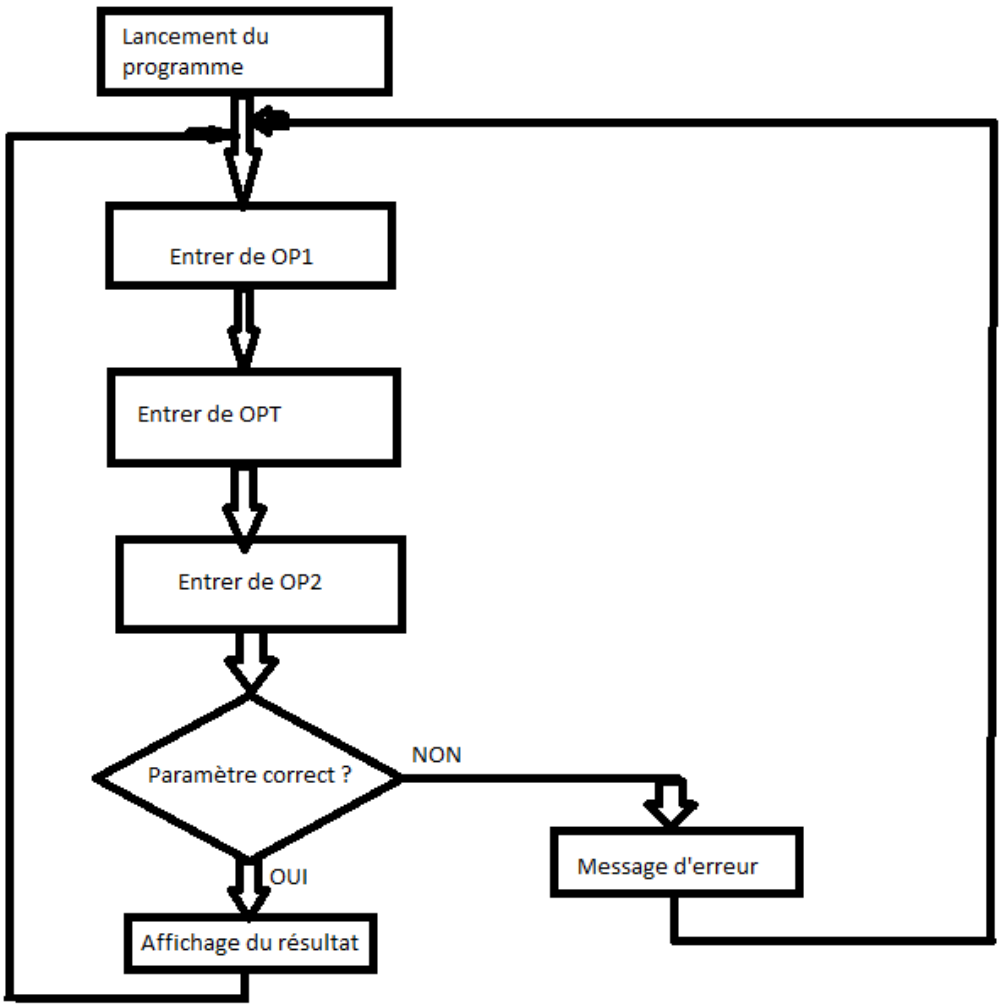


- Pour finir le programme, fermer les fenêtres ou faites Ctrl+C

2. Analyse

Pour concevoir ce socket, nous nous sommes basé sur les travaux réalisés en cours. C'est-à-dire que nous sommes restés sur une interface client-serveur. Le client va s'occuper de récupérer les données de l'opération que souhaite effectuer l'utilisateur. Ensuite, le client envoie ces données au serveur qui va traiter la demande et renvoyer le résultat. Pour l'envoie et la réception des données nous avons utilisés TCP étant donner que c'était la seul que nous avons vus jusqu'à présent.

Diagramme d'utilisation (IHM) :



3. Conception

Définition de la structure général du client :

```
Fonction main (
        récupération de l'adresse du serveur
        récupération du numéro de port
        création socket
        définition du buffer qui lit dans la socket
        définition du printer qui écrit dans la socket
        boucle (
                demande et recuperation du premier operande
                demande et recuperation de l'opérateur
                demande et recuperation du second operande
                envoie des donnees au serveur
                recuperation et affichage resultat
                )
        fermeture de la socket
```

Définition de la structure général du serveur :

```
Fonction main (
        récupération du numéro de port
        création de la socket serveur
        création d'une socket
        création du Buffer et du Printer
        boucle(
                récupération des donnees du Client
                gestion des exception, des mauvaises saisies,  si pas de problèmes
                        [calcul]
                sinon
                        [création message d'erreur]
                si message d'erreur existe
                        [envoie du message au client]
                sinon
                        [envoie du résultat au client]
                )
        fermeture des objets, des sockets
```

4. Problèmes rencontrés

- Même si pour commencer, nous nous sommes basé sur un modèle Client-Serveur déjà réalisé en cours, nous avons du apprendre à bien nous servir des BufferReader, PrintWriter, et commandes associés tels que ReadLine. Pour cela nous nous sommes appuyés sur le cours et des exemples trouvés sur Internet. La plus grosse difficulté a été de faire communiquer le client et le serveur. Dans cet exercice les méthodes OutputStream et InputStream n'étaient pas applicables.

- Le second problème a été la tentative de faire envoyé au serveur les opérandes au format Integer, alors que c'est impossible. On ne peut envoyer que des nombres au format String, c'est le serveur qui convertit ensuite au bon format. Cela nous a pris du temps à résoudre ce problème, car nous ne connaissions pas cette particularité.

- Afin de respecté le cahier des charges et proposer un programme qui ne s'arrête jamais, il fallait une boucle. Nous sommes parti sur un « tant que » avec une variable booléenne initialisée à « true » et jamais modifiée. Cependant le compilateur à détectée la boucle infinie et n'a pas créer les jar. Nous avons contourné le système en faisant un While(i==0), avec i initialisée à 0 et jamais modifié.

- Pour finir il y a eu la phase de test, permettant de trouver les différentes exceptions. Cette partie n'a pas été un réel problème, mais nous a demandé du temps et de la réflexion pour ne rien oublier. Nous avons notamment gérés les cas suivants grâce a un renvoie de message d'erreur :

Opérateur différents de + ; - ; * ; /

```
-----Bienvenue dans la Socket-Calculatrice-----
Entrez le premier operande :
2
Entrez l'operateur (+,-,/ ou *) :
j
Entrez votre second operande
2

 Le resultat :

2 j 2 = Operateur incorrect, entrer +, -, * ou /
```

Conversion impossible des nombres

```
-----Bienvenue dans la Socket-Calculatrice-----
Entrez le premier operande :
J
Entrez l'operateur (+,-,/ ou *) :
+
Entrez votre second operande
2

 Le resultat :

J + 2 = Conversion Impossible des operandes
```

Division par zéro impossible

```
-----Bienvenue dans la Socket-Calculatrice-----
Entrez le premier operande :
5
Entrez l'operateur (+,-,/ ou *) :
/
Entrez votre second operande
0

 Le resultat :

5 / 0 = Divion par zero est impossible
```

- Pour améliorer le programme nous avons pensé mettre au point un système avec un Switch-case pour le choix de l'opérateur mais manque de temps nous n'avons pas pu inclure cette amélioration dans le programme final. Nous nous sommes contentés de « If » imbriqués.

5. Bilan

Ce projet nous a permis d'appréhender les différentes difficultés qui peuvent se poser lors d'un projet informatique. Nous avons aussi pu mettre en pratique toutes les étapes qui constituent le développement d'une application client-serveur. Nous avons notamment appris à utiliser les BufferReader et PrintWriter.

Nous sommes restés sur une interface rudimentaire, mais le programme peut encore être amélioré. Par exemple avec l'intégration d'un Switch-case pour le choix de l'opérateur afin d'augmenté la clarté du code ou la création d'une interface graphique.